On the users' critical mass for smart city crowdsourcing applications

Dimosthenis Markopoulos

SID: 3301140000

Supervisor: Dr. Merkouris Karaliopoulos

SCHOOL OF SCIENCE & TECHNOLOGY

A thesis submitted for the degree of

Master of Science (MSc) in Information and Communication Systems

OCTOBER 2015

THESSALONIKI – GREECE

Abstract

This dissertation was written as a part of the MSc in ICT Systems at the International Hellenic University.

Dimosthenis Markopoulos

1/9/2015

Contents

1 Background / Motivation

This project touches on several concepts and paradigms, which have only emerged over the recent years and are far from common knowledge. Explicating these terms and summarizing related work on them is a prerequisite for explaining the motivation of the project and stating its objectives. The discussion that follows is not meant to be exhaustive; it rather seeks to position the project in the broader research arena.

1.1 Crowdsourcing

Estelles & Gonzalez [1] define crowdsourcing as "a typically online activity, where a group of individuals voluntarily undertake a task". The task at hand may be proposed to them by other individuals or companies. A key aspect in the conception of crowdsourcing is the possibility for *mutual benefit*: the task undertakers provide resources such as work, time or money. In return, they might receive monetary or non-monetary rewards, such as social recognition or perhaps self-esteem from the very act of contribution. Originally, another characteristic of crowdsourcing [1] has been that the contributors are typically acquired through open calls: anyone can choose to participate in a crowdsourcing venture, not just communities of specific interests or background, as one might imagine.

Despite the definition above, there seems to be a fuzzier line as to what constitutes crowdsourcing in real life. For example, there *do exist* services that target a specific subset of people but still identify themselves as being instances of crowdsourcing. An instance of that "breach" is *Freelancer.com,* where programmers can bid on projects, or *InnoCentive.com*, which targets professionals who essentially constitute an outsourced form of the R&D department of a company.

Complying more with the traditional crowdsourcing definition*, Mechanical Turk* (MT) is a popular platform built by the famous auction website Amazon.com. Introduced in 2005 [29], this service targets businesses and developers who in turn provide a list of tasks to potential contributors to choose from. These are typically simple repetitive tasks that can be remotely completed by the contributor in front of his own computer.

Examples [9] include “transcribing short media files to text” or “describing an image with up to 10 letters”. Tasks are usually paid, typically ranging from 5 to 20 dollar cents per task, although unpaid ones do exist as well. Each task has a time frame in which it can be completed and may require specific “qualifications” [10]. These are dynamically updated for contributors each time they complete a task and may reflect their reputation and performance on specific activities. Statistics collected [11] show that for the year of 2015, tasks available at any time in Amazon MT amount to an average of 200.000, with 80% of the subscribers/contributors being from the US. Other known examples of crowdsourcing applications include Wikipedia [18], OpenStreetMap [19], Yahoo Answers [20] and UrbanDictionary [21].On the other hand, *Task Rabbit* takes the crowdsourcing paradigm in mobile settings. This is a service targeting people who need assistance with doing something. Hence, a task could consist in helping someone assemble an IKEA piece of furniture, taking care of a pet while the owner is on holidays, or having something delivered somewhere. A set of prerequisites may also be set by the task requestor, such as a truck to carry something big or, more subjectively, “someone strong enough to carry a couch”. As before, a rating system is in place, helping decide who to choose when assigning a task. This is perhaps a crucial aspect, as personal safety and theft may come into play when dealing face-to-face with a stranger. It could also be part of the reasons why in 2011, just 23 people out of the 1500-count workforce were found to have completed a third of all tasks [12].

1.2 Crowdsensing

Crowdsensing can be seen as a particular instance of crowdsourcing in mobile settings. It becomes all the more relevant with the emergence and popularity of modern mobile phones, which include a multitude of sensors, such as camera, microphone, GPS and barometer. Task undertakers move around, making use of one or more sensors, whose data is being fed back to a centralized database. Ganti, Ye & Lei [2] break down crowdsensing applications into three types: environmental, infrastructure and social.

In the first case, natural environment attributes are measured, such as the weather or noise levels. In the second case, man-made infrastructure is monitored. For example, potential applications could be tracking road traffic or mapping free Wi-Fi hotspots across a city. Finally, in the third case, individuals share information about themselves

rather than the external world. A movie-rating app, or sharing and comparison of fitness exercise data could be seen as two examples of social crowdsensing apps.

Privacy, security and data integrity, count as major challenges in a mobile crowdsensing environment [2]: to measures are required to protect the privacy of users and ensure that submitted data has not been falsified. At the same time, these aspects need to be addressed in a manner that preserves the functionality of the application under consideration.

Another major challenge in crowdsensing applications is the "proper" selection of user-sensors. Many users may be able to provide the same or highly similar data, so a way has to be found to choose which user will be selected. Several suggestions [3] are listed below for assessing/ranking potential contributors and their data in this respect:

- Reputation/Credibility:
  A user is selected according to a reliability factor, such as the average rating of his/her past contributions by other users.
- Quality of data the user is able to contribute:
  For example, one user might have a mobile phone with a better camera than another, so for a photo-based application he will be deemed as a better candidate.
- Cost:
  For example, in a monetary compensation scenario, one user might charge less for the same contribution than others.
- Task coverage ability:
  A set of users is selected so that a maximal number of completed tasks is achieved (or the same task, involving many contributions at different locations and time epochs, is more fully completed).

Depending on the optimization objective, the resulting problem may lend to different instances (or variations thereof) of optimization problems. In the last case, the problem could be formulated as an instance of the *Maximum Coverage problem* {ref}. For example, in a hypothetical terrain mapping scenario, we can think of space as a grid of tiles, within which users move. A user is considered to be able to cover a tile as long as (s)he exceeds a threshold of time spent on it. This way, each user may be assumed to be able to cover a set of tiles.

Then, the optimal solution to the problem implies selecting the minimum number of users who can cumulatively cover a tile set. This is an NP-complete problem and it has been shown that the greedy algorithm achieves the best approximation ratio for it [4]. Essentially, this means that, first, the user with the highest number of covered tiles is selected; then, the user with the highest number of tiles not yet covered is chosen; and so on. The selection of users/sensors continues with this rule until all tiles have been covered.

Generally, other applications could also work in the same manner. We could for example, instead of terrain segments, allocate a set of tasks to each user and, similarly, find the minimum number of users to cover all tasks.

Finally, much effort has been devoted to the design and realization of incentive mechanisms that could ensure the engagement and quality contributions of end users to crowdsensing applications. In general, the incentives for the participation of a mobile user may be (a) monetary, as in [30], where participants bid their prices and sell their own sensing data to the platform; (b) non-monetary as in [31], where people volunteer to perform spatial tasks without expecting any reward; and (c) a combination of both, as in [32], where a combination of small monetary payments with gamification techniques is proposed.

Experiments with online labor markets in 2009 [33] report that increased financial incentives increase the quantity, but not the quality, of crowdsourced work. To ensure quality, incentives need to be combined with the proper recruitment/selection of users, as outlined earlier. In [34], for instance, authors want to incentivize users to perform online tagging tasks. Their goal is to maximize the quality of tagged data which is defined as the number of contributions gathered by the users. On a similar aspect, in [35] the aim is to maximize the utility of their system, as reflected in the number of completed assignments.

1.3 Open data/Civic data

We inarguably live in an age where a significant amount of data is constantly generated and stored in databases. This applies to various sectors, such as sciences (e.g. DNA analysis in biology) and government (e.g. statistical data about citizens). The "open data" movement tries to make certain data, such as the above, available for anyone to use without any restrictions or forms of control [5].

The Open Data Index [13] ranks key national statistics (such as demographic and economic indicators), as well as government budget information as the most widely available datasets globally. The availability of such data, however, is far from uniform across the world. The African continent as well as western Asia fall understandably behind the rest of the world in this respect, in what seems to be another demonstration and/or consequence of digital divide. As perhaps expected, the US, Australia and Western Europe rank top regarding the wealth and quality of available information. In the case of Greece however, we see that both types of datasets are available but not openly licensed.

Arguments have been placed both *for* and *against* open data. Proponents typically argue that certain data belongs to the human race as a whole and that their free availability promotes and preserves scientific research and social welfare. On the other side, among others, it is pointed out that the cost of collecting and maintaining data must be somehow reimbursed.

Civic data is already being collected and utilized in major cities. A popular example of such data is information about the city infrastructures, such as subway routes and the location of various points of interest (POIs): hotels, gas stations, famous landmarks or other places in the city that are assumed to be of high interest for a citizen or a tourist. Another instance of civic data are statistical data such as about the city traffic levels and crime rate per city area. It seems then that civic data would be a good candidate for being open. Indeed, *experimental* applications have been developed also in Greece that make use of existing open civic data, sometimes combined with a crowdsourcing aspect [6]. An example is a route-sharing application for sightseeing, where tourists can post their own paths and get suggestions that fit their profile. In another application, people with mobility problems can define a destination and a time frame, so that if it coincides with a volunteer's schedule, they can both carpool to their destination.

1.4 Smart city/Smart transportation

A formal definition of the "smart city" concept refers to the use of digital technology to essentially enhance its citizens' daily life and processes within it. A key characteristic is the use of data combined with AI (artificial intelligence), to make efficient use of its physical infrastructure [7]. A smart city is considered to be able to efficiently adapt to a changing environment. It places emphasis on its citizens to achieve this, who, on a broader scale, also take part in designing it [8].

Many definitions of "smart city" fail to mention two other key aspects of it: crowdsensing and open data. This is perhaps on purpose, as it is a broad-scope vision, open to many interpretations. As a result, "data" is used as a blanket term to refer to any input needed. On the other side, most popular working implementations rely on these two means of acquiring data. One could point out that technically, open data is an initiative, not a service paradigm like crowdsensing. Both are similar however in their being "enablers" for many applications to work.

Many ideas have been proposed, such as electricity sharing or solar roadways, however these are usually just a concept or, at most, in the start-up phase. A possible smart city example could involve lampposts. The smartness in this case relates to both their electricity consumption and their enhanced functionality. On the first front, of the lamps do not adhere to a fixed on-off cycle but are equipped with sensors that can be used to turn on the light even in the daytime, depending on conditions such as fog or heavy rain. Regarding the additional functionality, one could mention the mesh-connected lampposts that have been put in place in the cities of Chicago and Philadelphia [14], integrating traffic direction, flood/earthquake detection, as well as broadcasting emergency messages to pedestrians.

However, not all smart solutions imply the presence of physical sensors. In an unusual instance, London uses predictive modelling to prevent blockages in its sewer network [15]. By analyzing the network topology, weather conditions and age/material of pipes, high-risk sewers can be identified and monitored, to prevent incidents, as was the case in 2013, when a "bus-sized" lump of fat was identified and removed [16].

Smart transportation solutions are viewed as integral parts of any smart city implementation. For a typical commuter, an intuitive definition would be using technology to minimize time spent getting to a destination. Others might cite the cost or even the environmental impact as more important factors. A notable example is the

traffic monitoring application *Waze*: it combines static map data with mobile phone sensor data to compute the fastest routes in real-time and report any incidents. Users can also place "pins" on the map, signifying speed cameras, traffic accidents, etc.

For the city, smart transportation might translate into proactively designing infrastructure to accommodate current and future needs. The city of Vienna for example has developed car-free complexes [17], served by public transportation in an efficient manner. Space saved by not having garages is used to build community facilities and landscaping, "boosting the quality of life for residents of the complex". Real-time events could also be monitored and affected accordingly. For example, traffic sensors placed on the roads combined with signs can be used to suggest faster routes depending on traffic conditions. Or (semi) real-time maps of parking space availability may help better directing car traffic in dense urban areas and avoiding the cost of needless cruising across the busy urban roads.

1.5 The focus of our work

The Smart City term points to a really broad vision that addresses various aspects of every day city operations and its citizens' activities. Whereas the actual limits and the sustainability of this vision are issues open to discussion, its "ingredients", *i.e.*, the fundamental enablers of this vision have become clearer over time. And one thing that has become a certainty over these years is that data availability stands amongst the top prerequisites in the list. Where are these data to come from?

Primarily from two sources, and this is where the open data initiative and the crowdsensing service paradigm become most relevant. None of them can be taken unconditionally for granted, for different reasons. With respect to making civic data open, one needs to secure the commitment of major institutions at city (e.g., municipality) but also at national level (e.g., police/ministries). Part of this exercise is to address the questions about the cost of storage, maintenance, and presentation of collected data. Therefore, their availability has more to do with policy decisions at different levels of administration.

On the other hand, a significant amount of data can be contributed by individual citizens through the crowdsensing mechanism. In several cases (e.g., Waze, others), the technology has shown its potential to generate a service that can assist the city

operations in tangible manner. However, the grand challenge here, besides addressing privacy concerns, is to recruit and sustain the necessary volumes of contributors ("the critical mass") that can render the service acceptably operational. To this end, various incentives have to be given to end users, in line with their heterogeneous preferences and motives for contributing data to the different apps.

The actual share of civic vs. crowdsourced data in the final mix cannot be predicted at the moment and depends heavily on the application at hand. However, in most cases, the two sources can complement each other, improving the quality of the final service offering.

In this work, we take a generic approach to the study of city crowdsensing applications, having in mind primarily smart transportation solutions. Our motivating remark is that before designing incentives for attracting crowds, we should have first gain some understanding about the number of contributors needed to provide a satisfactory service. Likewise, if open data is needed, it should be investigated whether this exists in a usable form by public/private institutions – and if it does, verify that it is openly licenced for use.

In the remaining part of this Thesis, we will firstly present our modeling framework for a generic crowdsensing application and the way we accommodate some to-be-open civic data (Chapter 2). Subsequently, we analyze the simulation tool that will be used as to its parameters, output metrics and performance (Chapter 3). Next, the simulator will be used on a real-world map to acquire sensitivity information (Chapter 4). Finally conclusions will be drawn and further relevant work might be suggested (Chapter 5).

2 Problem Definition / Methodology

Urban transportation has been one of the main fields that is expected to benefit from crowdsensing technologies. Several smart-transportation applications can be realized through with the help of the crowds and most of them consist in retrieving data from specific points/locations across the city. For example, for an application that tries to trace the coherence of buses to their schedule the natural points of interest, where information is generated, are the bus stops. Likewise, in an application that seeks to give drivers assistance by their search for parking, the natural points/locations of interest (PoIs) are the individual on street parking spots across the city center.

The kind of information that is contributed in each application also varies. Therefore, it may consist of continuous values, representing time lag, in the case of bus schedule monitoring applications; or binary, when trying to track the availability status of a parking spot.

Hence, the approach we intend to take in our study is deliberately generic in that these applications will be abstracted to their maximum possible common denominator. The aim is to derive conclusions and insights that have a broader scope.

2.1 System model

In the modelling scheme that will be used, the application is abstracted as a user requirement for a specific PoI (Point of Interest) sampling delay. For example, an application may give us a user requirement to achieve a specific sampling delay value for e.g. 50% & 95% of times.

The reference action space is an urban area A of size S. Scattered across this area are L Points of Interest (PoIs), whose context depends on the specific application (e.g., parking stops, bus stops etc). For reasons that will become apparent later, this area may be split into a number of cells, each one containing zero, one or more PoIs.

At any point in time, these PoIs occupy one of a finite number of states. For example, a parking spot can be in one of two states, occupied or spare; the time-to-next bus arrival at the LED displays at bus stops takes continue values within a closed interval, [0, B], where B an upper bound to the residual time till a bus arrives (equal to the time spacing of the routes of successive buses under normal conditions). PoIs alternate between states, either in fixed intervals or randomly.

We also consider a population of N agents, be it pedestrians or cars moving around this area. The mobility pattern may vary broadly: some of these agents may be traversing the area A, others may be moving all over it and some others may only roam within a certain subarea of A, at different speeds. The agent population may represent the full city population or its subset owning a smartphone and sets an optimistic upper bound to the number of subscribers to the application, In practice, we will split this N users into three subsets: those who have not even downloaded and subscribed to the application, those N1 who have and are always online contributing data whenever they are within reach of a PoI, and those N2 who occasionally make contributions because they are intermittently online or because they are less committed to the application objectives and ambitions. Apparently, the subscriber base of the application equals N1+N2. The agents are assumed to be moving between these cells with some predefined model. These models will describe different movement speeds as well as time spent on a tile. In addition, they may describe agent groups which favor specific patterns, such as the perimeter or the center of the area S. The baseline model features a matrix M= CxC, where C is the number of cells in A, each element M(i,j) describing the rate at which an agent moves from cell i to cell j. The probability of transition to a neighboring tile may

be equally distributed among neighbors or, perhaps, we could place more weight on specific pattern-forming tiles to coarsely model hotspots or high-traffic roads. Each time a user finds himself in a tile, (s)he is assumed to be able to collect and give data about the PoIs of the particular tile.

2.2 Evaluation methodology

Input to the evaluation process will be the following:

- The size of the subscriber base of the application (numbers N1 and N2). These are going to be the free parameters in the evaluation. The main objective of the evaluation will be the sensitivity of performance to this quantity.
- The number and location of PoIs. Our ambition is to get information about real PoIs in the city of Thessaloniki. Such data are, for instance, parking maps, maps of bus stations across the city and, ideally, would be made available electronically under the Open Data initiative.
- The mobility patterns of agents, including trajectories of their movement and speed. Modeling-wise, these could be equivalent to specifying the matrices Mu, $1 \leq u \leq N$ and cell residence time for each agent u.

The application performance for given subscriber base (N1, N2), mobility patterns and number of PoIs L will be measured by application-specific metrics. For the parking application, the key performance indicator will be how quickly it responds to changes in the availability status of parking spots. Specifically, the ultimate goal would be to find the mean time elapsed from the point a spot gets vacated to the time this gets "sensed" and reported to the system.

3 Simulation

The simulator has been developed in full within the MATLAB environment, version R2015a for Windows. The written code addresses three main components: the physical space layout, the mobility of the crowd (users) and the emulation of the sampling process of the different PoIs by the moving users.

3.1 Modeling constructs

More specifically, the following components have been implemented in the simulator:

3.1.1 Physical space layout

In the simulator, the area of interest is modeled as a rectangular matrix of cells (grid). This is the area, within which users move and the PoIs that have to be monitored are located. Each cell in the grid has its unique coordinates (x,y) corresponding to the row and column it occupies in the grid. When mapping the real world, each cell may correspond to a road segment of some (10s of) meters or some building block. Each cell may contain a different number of PoIs, also depending on the application at hand. For example, in the parking application case, PoIs (*i.e.,* parking spots) are encountered in cells that encompass on-street parking spots (road segments) and parking lots. On the contrary, cells that map buildings do not include any PoIs.

The size of cells presents a tradeoff between accuracy in modeling the user mobility and sampling process (ref. section 3.1.2) and complexity. Namely, larger cells concentrate more PoIs and make the user mobility patterns coarser. As the cell size gets smaller, the modeling accuracy of simulator increases but more calculations and state are needed that increase the simulation run times.

To produce the number of PoIs per cell, a 20x20 cell grid was overlaid on the original map, as shown below. (The choice of this value is explained in 3.1.2) For each cell, it was then calculated how many PoIs it covers. Given that the map provides total PoIs per road segment, it was assumed that these are evenly spread throughout that segment.

Some streets may only have a few PoIs, which means that some of their smaller segments may end up having zero PoIs allocated. This would be different from densely populated streets, where every segment has at least 1 PoI.

Furthermore, in the "Conclusions" chapter of this project it is discussed how other real-life maps, of same or different structure, can be potentially used in the simulator.

Figure 1: Grid overlay on map to produce cell POI number

3.1.2 Sampling process

The PoIs are sampled by users while they move or pause within the cell that hosts these PoIs – the last samples essentially being taken the moment before departing from a cell and within the sampling rate allowance of the application. The PoIs sampled are strictly the ones contained in that specific cell. It makes sense then that an appropriate cell size should be chosen, so that a pedestrian has the ability to observe all POIs within it. In the actual implementation, potential contributors of the application are assumed to be able to obtain and report all PoIs within a 15 meter radius. This translates to a cell size of

roughly 30x30 meters. The area chosen to be modeled is of size 600x600 m^2, thus a matrix of size 20x20 cells emerges.

3.1.3 Execution mode

The simulator can be run in two ways. One option is to run it for a specific set of input parameters. The other option is to run it as a batch operation, with different sets of inputs automatically being inserted as parameters.

In both cases, when the simulator is run, users start moving within the cell matrix. By the end of the simulation, for each cell, it is recorded how many times users have landed on it, and when this happened. This information is processed to produce a list of intervals between successive sampling instances for each PoI, as well as its average interval value.

Single input

This option may be used to test the simulator, as well as get a rough approximation of results for a specific input. It also gives the ability to track the movement of users in detail and get statistics for every cell. These get exported to text files, a sample of which is shown below:

```
Cell 13 19 has 4 POIs, updated 3 times with an average delay of 199.33
Cell 14 14 has 3 POIs, updated 6 times with an average delay of 129.00
Cell 14 15 has 6 POIs, updated 7 times with an average delay of 111.71
Cell 14 16 has 4 POIs, updated 5 times with an average delay of 162.00
Cell 14 17 has 11 POIs, updated 1 times with an average delay of 123.00
Cell 14 18 has 1 POIs, updated 3 times with an average delay of 268.00
Cell 14 19 has 6 POIs, updated 2 times with an average delay of 266.00
Cell 15 10 has 6 POIs, updated 13 times with an average delay of 59.62
```

Figure 2: Cell statistics text file

```
User 9 moving from 9 16 heading towards 9 19 - via cell 9 17.
---User 10 leaving destination 16 20, new destination is 3 4.---
User 10 moving from 16 20 heading towards 3 4 - via cell 15 19.
User 1 moving from 17 14 heading towards 17 19 - via cell 17 15.
User 2 moving from 5 17 heading towards 11 10 - via cell 6 16.
---User 12 leaving destination 17 19, new destination is 2 8.---
User 12 moving from 17 19 heading towards 2 8 - via cell 16 18.
User 7 moving from 19 10 heading towards 14 10 - via cell 18 10.
User 9 moving from 9 17 heading towards 9 19 - via cell 9 18.
```

Figure 3: User movement text file

The distribution of all interval values is also plotted against time in normal and CDF form, along with the distribution of average delay for every PoI.

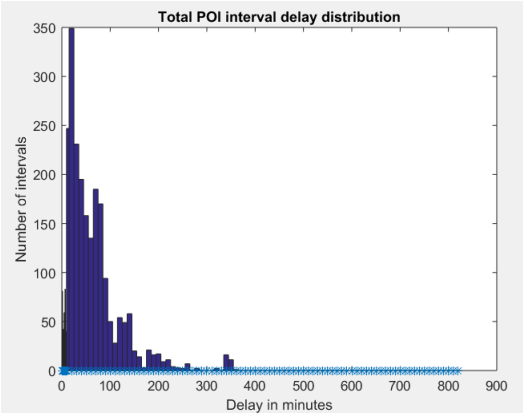

Figure 4: Delay distribution plot

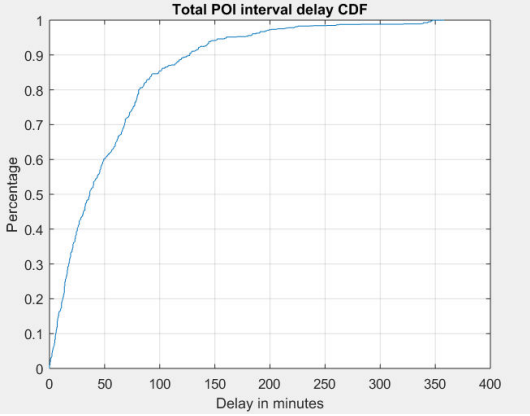

Figure 5: Delay distribution CDF plot

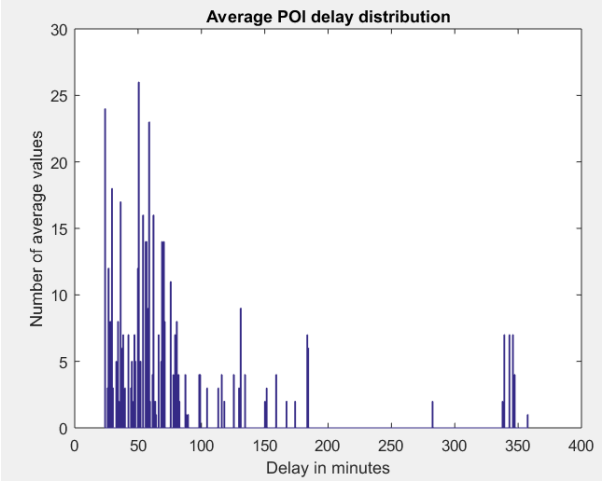

Figure 6: Average delay distribution plot

Batch input

With this option, we can compare and plot results for different input values. The goal is to infer the critical number of users needed to support specific PoI sampling delay values. To achieve this, every time a set of parameters is tested, the 50^{th}, 75^{th} and 90^{th} percentile values are extracted from the total PoI sampling delay distribution. For accuracy, the simulation is repeated a user-set number of times for a specific input. Then, value averages are computed, along with the 95% confidence interval bounds. It

should be noted here that a separate MATLAB function was obtained [22], which enables horizontal plotting of confidence intervals.

The above process gets repeated for all inputs. The final executable specifies the simulator input, which is range of user number values, a range of mean residence times for users and simulation running times. Ultimately, results get plotted against user number. Three plots are generated (for each percentile value) with average values along with confidence bounds, as shown below. Each plot contains overlaid "curves" for different residence time values.

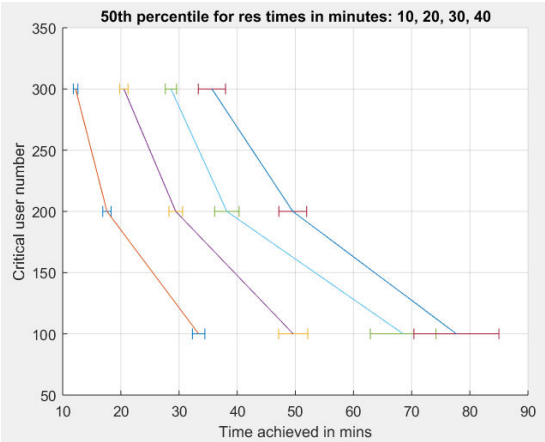

Figure 7: Output plot for 50th percentile

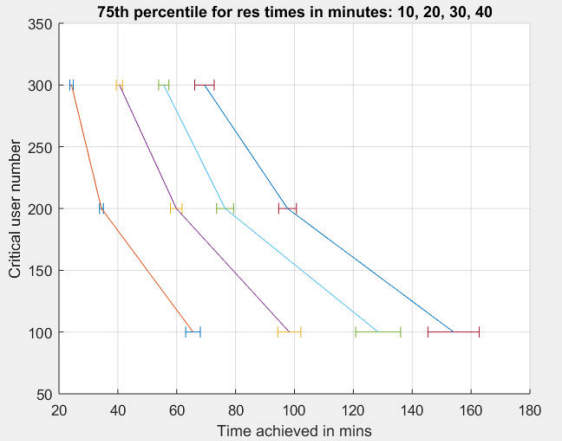

Figure 8: Output plot for 75th percentile

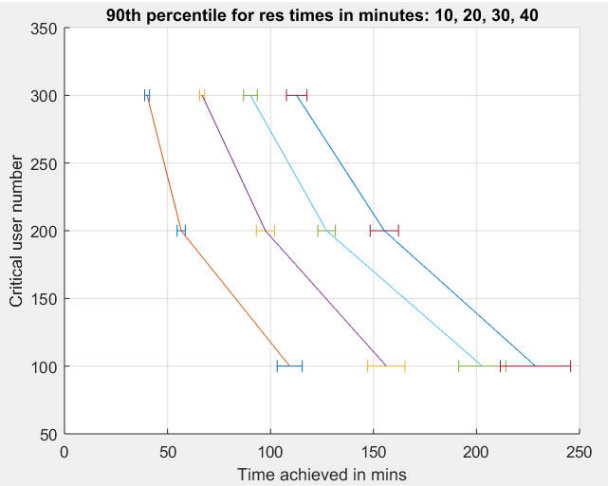

Figure 9: Output plot for 90th percentile

3.1.4 User movement

In the simulation runs reported here, individual users move across the grid of cells according to random mobility models. The baseline mobility model is based on the "Random Waypoint (RWP)" model [28]. It is essentially an adaptation of the continuous RWP model over a grid. A user starts from a random cell on the grid and after staying there for some time $t_{pause,}$ (s)he chooses another random destination cell on the grid. (S)he then moves towards it at walking speed, v_{walk}, in a hop-by-hop fashion, each time choosing the best neighboring cell diagonally, horizontally or vertically. Especially when diagonal movement is relevant, the latter is terminated and the movement becomes horizontal or vertical, when the current coordinates become equal to one of the two destination cell coordinates. Once the destination is reached, the user again chooses some time to spend there, before selecting the next destination and moving towards it..

For practical purposes, the population of users within a single simulation run is static, and their movement is confined within the map boundaries. It can be argued nonetheless that both choices are compatible with realistic movement patterns: Firstly, the population is meant to represent an average population over some time, not at any specific time point. Secondly, one can imagine that the fixed user population moving on the map corresponds to different individuals over different time intervals. In real life, users could be walking in and out of the simulation area (map),while their number *on average* remains stable.

The mobility model can be parameterized with respect to:

a) **the cell pause (residence) time** over which users pause their movement when they reach their destination cells. We have considered exponentially distributions with varying mean values in mins. The effect of different pause times on performance can be better seen in the batch execution mode, which produces overlaid graphs for every value.

b) **their average walking speed** as they traverse cells, on their way to their destinations. We have considered contributors to be pedestrians only, whose average speed [23] is 1.4 m/s. Also, slower/higher speeds were tested.

To model a specific speed, we have to adjust a variable which denotes the ratio of simulator time to real time. The two factors that should be considered when choosing a ratio value is the cell size, as well as the average time needed by users to traverse a cell.

In our case, cells are 30x30 meters long. For a speed of 1.4 m/s, 21.4 seconds are needed to cross a cell horizontally/vertically, and 30.2 seconds to cross diagonally. This gives an average traversal speed of 26 seconds in the course of the simulation. As mentioned before, movement occurs step-by-step, where in each simulator iteration, users hop from cell to cell. If cell traversal is 26 seconds (1 iteration), 1 minute is **2.3** iterations. This final value acts as an input to the simulator and can be adjusted to model different walking speeds for a specific cell size.

c) **the way the destination cells of their movement are chosen.** We have implemented two scenarios in this respect. In the first one, hereafter called "random destinations" users choose randomly their destination cell. It has been shown in [24] that the steady-state distribution of user across the area of movement tends to be bell-shaped around its center, i.e., users are concentrated in the more central areas and their number declines monotonically as we move towards the area edges. The second scenario ("hotspots") includes hotspot areas that serve as attractors for users and shape their mobility accordingly. Namely, the choice of destination cells is biased towards cells lying in one of these hotspots.

To evaluate the impact of hotspot preference, we can adjust a variable that acts as input to the simulator. This defines the probability of users choosing a random hotspot cell (from a user-defined list) as their next destination.

In our case, 18 cells have been determined to be the busiest in the map. These comprise streets "Milioni" and "Tsakalof", as well as a portion of "Skoufa" street.

3.1.5 PoI allocation

Besides the original PoI allocation, the simulator gives an option to reallocate PoIs randomly before the simulation, to evaluate the performance of different PoI distribution across a given topology.

In our case, two zones have been defined: the center and the perimeter. The center is considered to be the central ¼ of the map (the square formed by cell 6,6 as bottom-left corner and cell 15,15 at the right-top corner). The remaining cells constitute the perimeter area.

We consider two scenarios for the reallocation of PoIs--): in the first case, all central PoIs are scanned and stochastically moved to a random place in the perimeter. In the second case, it is perimeter PoIs that (may) move towards the center. The probability value for both cases can be adjusted and serves as input to the simulator.

3.1.6 Variables

Summarizing the parameters that can be tweaked for the simulator, these are:

Variable	**Description**	**Name**
Grid size	This specifies the range of movement of users and should coincide with the size of the square matrix used to model an area.	Gridsize
Pedestrian speed	The value of the speed of pedestrians. This is defined by the relation of real time to simulator time.	time_ratio
Mean residence time (or times in batch mode)	The time users spend on a cell before moving on to another one.	mean_residence_time
Hotspot preference probability	The probability of users choosing a hotspot as their next destination.	p_freq
Reallocation probability	The probability of PoIs to relocate	p_move

	in the map, in a random place within a predefined zone.	
User number (or numbers in batch mode)	The number of users moving within the map.	Nusers (in single mode) user_numbers (in batch mode)
Number of runs (when in batch mode)	Times the simulator will repeat for a specific input, to compute an average output.	run_times
Simulation time	This is the time, in minutes, the simulation will run for. It should typically be 10 times more than the maximum mean residence time, to produce accurate results.	sim_time

Table 1: List of simulator variables

3.2 Performance metrics

As mentioned in chapter 3.1.3, the basic function of the simulator is to record the delay between successive sampling instances of PoIs. This information forms the basis for producing performance metrics.

The goal of the simulator is to find the user number requirement to support various crowdsourcing applications, depending on their sampling delay restrictions. Thus, the main performance metric is the required number of users, i.e. what we call users' **critical mass, CM(d;α)** that can achieve a specific PoI sampling delay value, d, for α% of time .

In the simulator, the number of users serves as an input, whereas the PoI sampling delay is the output. This means we cannot a priori set a specific PoI sampling delay value and compute the corresponding critical mass. Instead, multiple values for the user population should be tested to find where the desired delay is achieved, as well as determine the slope of the critical mass graph, i.e., its sensitivity to the user population

variations. Running the simulator in batch mode gives us this possibility, also letting us draw the confidence intervals of the computed values.

3.3 Validation of the simulator modules

Before initiating the experimentation with the simulator and the derivation of the aforementioned metrics, we designed and run some test scenarios with the aim to validate the developed modules.

3.3.1 User movement vs. PoI sampling process

The simulator was tested on the following scenario:

- Users follow the paths as depicted below.
- At time T=1 they start moving towards their destination. Each time unit corresponds to 1 movement step.
- When they finally reach the destination (at T=4) they stay for 2 time units.
- After that, they proceed to move towards a new destination (cell 20, 20).
- Simulation is run for 6 time units.

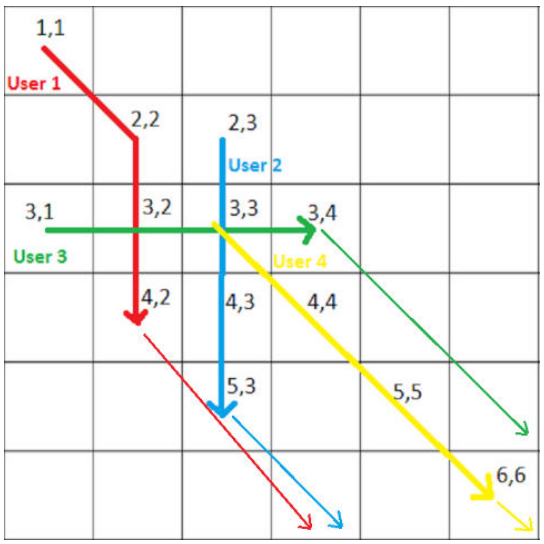

Figure 10: User mobility pattern

We can see that updates occur at:

- Cell 1,1 : T=1
- Cell 2,2 : T=2
- Cell 2,3 : T=1
- Cell 3,1 : T=1
- Cell 3,2 : T=2, T=3
- Cell 3,3 : T=1, T=2, T=3
- Cell 3,4 : T=6 (when user 3 leaves)
- Cell 4,2 : T=6 (when user 1 leaves)
- Cell 4,3 : T=3
- Cell 4,4 : T=2
- Cell 5,3 : T=6 (when user 2 leaves)
- Cell 5,5 : T=3
- Cell 6,6 : T=6 (when user 4 leaves)

The following screenshots show the simulator output, which is in line with the expected results.

```
Cell 1 1 has 0 POIs, updated 1 times with an average delay of 1.00

Cell 2 2 has 0 POIs, updated 1 times with an average delay of 2.00
Cell 2 3 has 0 POIs, updated 1 times with an average delay of 1.00

Cell 3 1 has 0 POIs, updated 1 times with an average delay of 1.00
Cell 3 2 has 0 POIs, updated 2 times with an average delay of 1.50
Cell 3 3 has 0 POIs, updated 3 times with an average delay of 1.00
Cell 3 4 has 0 POIs, updated 1 times with an average delay of 6.00

Cell 4 2 has 0 POIs, updated 1 times with an average delay of 6.00
Cell 4 3 has 0 POIs, updated 1 times with an average delay of 3.00
Cell 4 4 has 0 POIs, updated 1 times with an average delay of 2.00

Cell 5 3 has 2 POIs, updated 1 times with an average delay of 6.00
Cell 5 5 has 2 POIs, updated 1 times with an average delay of 3.00

Cell 6 6 has 0 POIs, updated 1 times with an average delay of 6.00
```

Figure 11: Cell statistics output file (edited to show only cells of interest)

```
User 1 moving from 1 1 heading towards 4 2 - via cell 2 2, at time= 1
User 2 moving from 2 3 heading towards 5 3 - via cell 3 3, at time= 1
User 3 moving from 3 1 heading towards 3 4 - via cell 3 2, at time= 1
User 4 moving from 3 3 heading towards 6 6 - via cell 4 4, at time= 1

User 1 moving from 2 2 heading towards 4 2 - via cell 3 2, at time= 2
User 2 moving from 3 3 heading towards 5 3 - via cell 4 3, at time= 2
User 3 moving from 3 2 heading towards 3 4 - via cell 3 3, at time= 2
User 4 moving from 4 4 heading towards 6 6 - via cell 5 5, at time= 2

User 1 moving from 3 2 heading towards 4 2 - via cell 4 2, at time= 3
User 2 moving from 4 3 heading towards 5 3 - via cell 5 3, at time= 3
User 3 moving from 3 3 heading towards 3 4 - via cell 3 4, at time= 3
User 4 moving from 5 5 heading towards 6 6 - via cell 6 6, at time= 3

---User 1 leaving destination 4 2, new destination is 20 20.---
User 1 moving from 4 2 heading towards 20 20 - via cell 5 3, at time= 6
---User 2 leaving destination 5 3, new destination is 20 20.---
User 2 moving from 5 3 heading towards 20 20 - via cell 6 4, at time= 6
---User 3 leaving destination 3 4, new destination is 20 20.---
User 3 moving from 3 4 heading towards 20 20 - via cell 4 5, at time= 6
---User 4 leaving destination 6 6, new destination is 20 20.---
User 4 moving from 6 6 heading towards 20 20 - via cell 7 7, at time= 6
```

Figure 12: User movement output file

3.3.2 Spatial node distribution vs. mobility patterns

To visualize user concentration, three tests were run. To produce an adequate number of cell updates, the simulator input was 50 users, with average residence time set to two minutes, while simulations were run for 1000 minutes. Note the prominent bell-shaped spatial node distribution when using true-RWP movement (figure 13), as described in chapter 3.1.4. The effect of selecting specific hotspot cells (figure 14) can be seen in Figure 15.

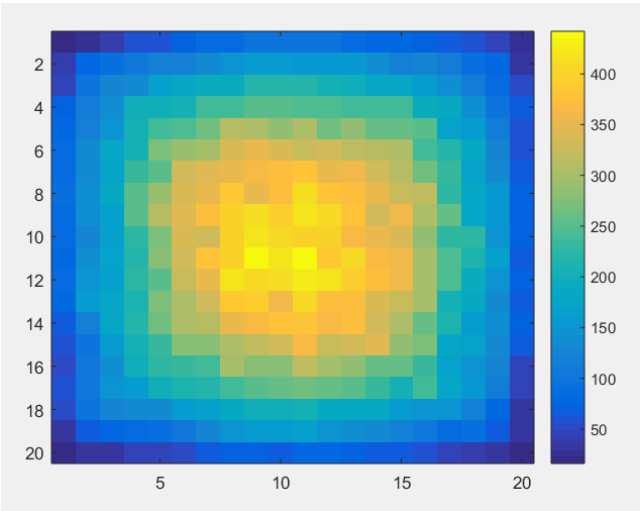

Figure 13: Cell update count for random destination

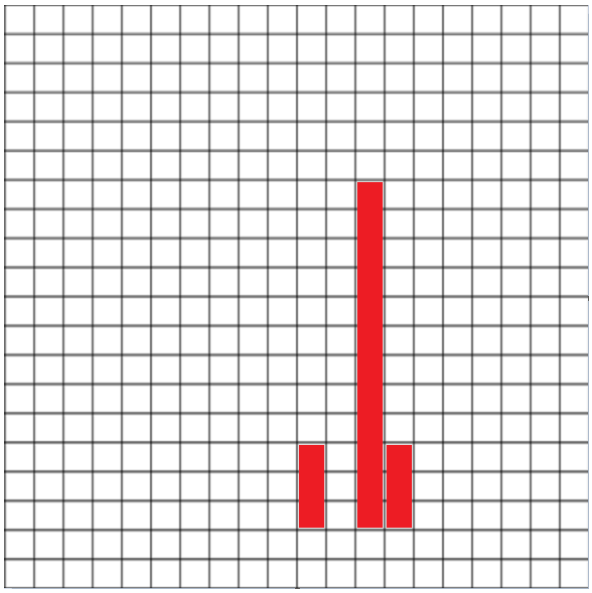

Figure 14: Hotspot cells

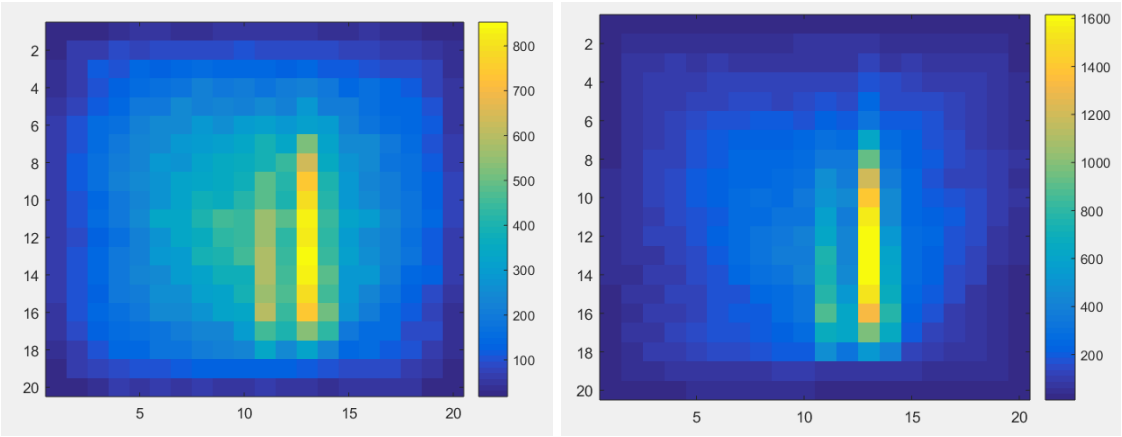

Figure 15: Cell update count for hotspot preference probability p_freq =25% (left) and 50% (right)

3.3.3 Simulation accuracy vs. number of runs and duration of simulation runs

There are two factors that tend to increase the accuracy of simulation results. These are the time the simulator is run for each scenario, as well as the number of times each scenario repeats itself to compute an average output value. In the following figures, note the reduction in error margins when any of these factors is increased.

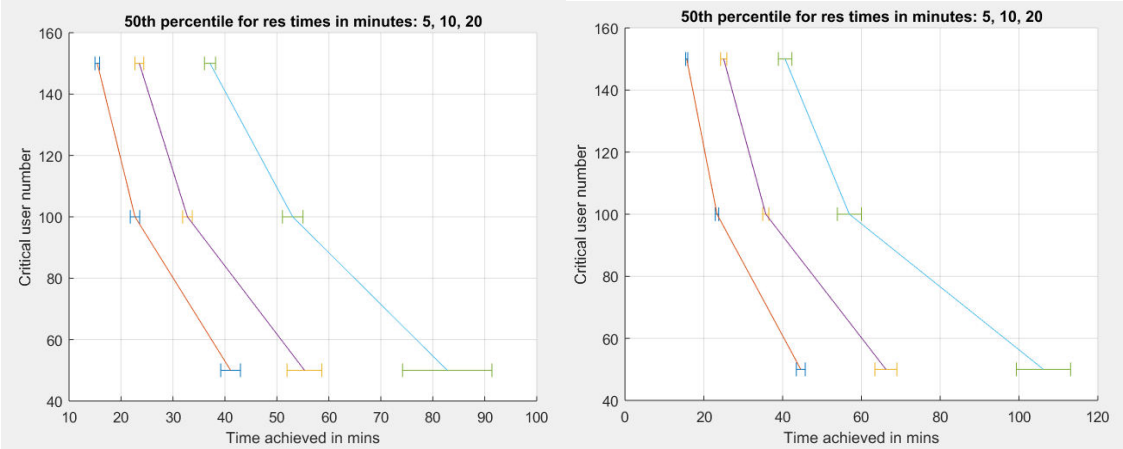

Figure 16: Obtained critical mass values when simulation time = 6 hours (left) and 12 hours (right)

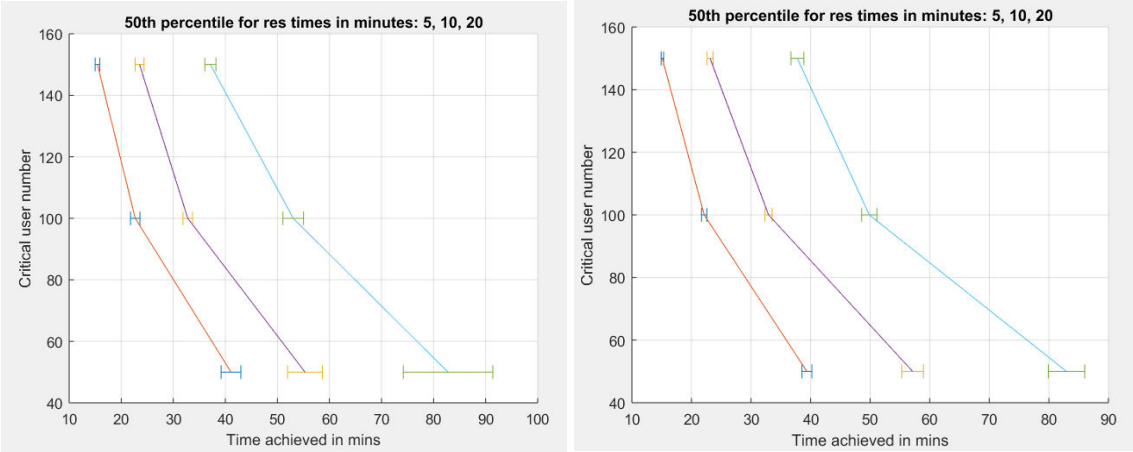

Figure 17: Obtained critical mass values when the number of runs equals 10 (left) and 30 (right).

4 Sensitivity analysis

In this chapter, the simulator is tested on the actual map of Kolonaki-Athens. The purpose is to get a feel for the critical number of users required to support specific PoI sampling delays in the context of a crowdsourcing application.

To compare results, the baseline scenario assumes that:

- Users move at 1.4 m/s
- Residence time is exponentially distributed with an average value of 20 minutes

Simulations were run 10 times, each one lasting 6 hours. Average values of 50^{th} & 90^{th} percentiles are plotted, along with 95% confidence intervals.

4.1 Impact of residence time

PoI sampling delay values, as expected, increase monotonically with the users' mean residence time. The time users pause their movement essentially corresponds to time that is "wasted" from the application point of view since users do not contribute to the PoI monitoring process.

The critical mass grows significantly as the application becomes more demanding: whereas 60 users suffice to achieve a PoI sampling delay of 5 mins over 50% of time, more than 210 are needed to achieve the same PoI sampling delay score over 90% of time. Likewise, as the application requirements become stricter, increases in the expected residence time raise more dramatically the critical mass numbers (from 60 to 100 and 130 for 5 mins median value vs. from 210 to approximately 400 and beyond 500 for 5 mins 90th percentile value.

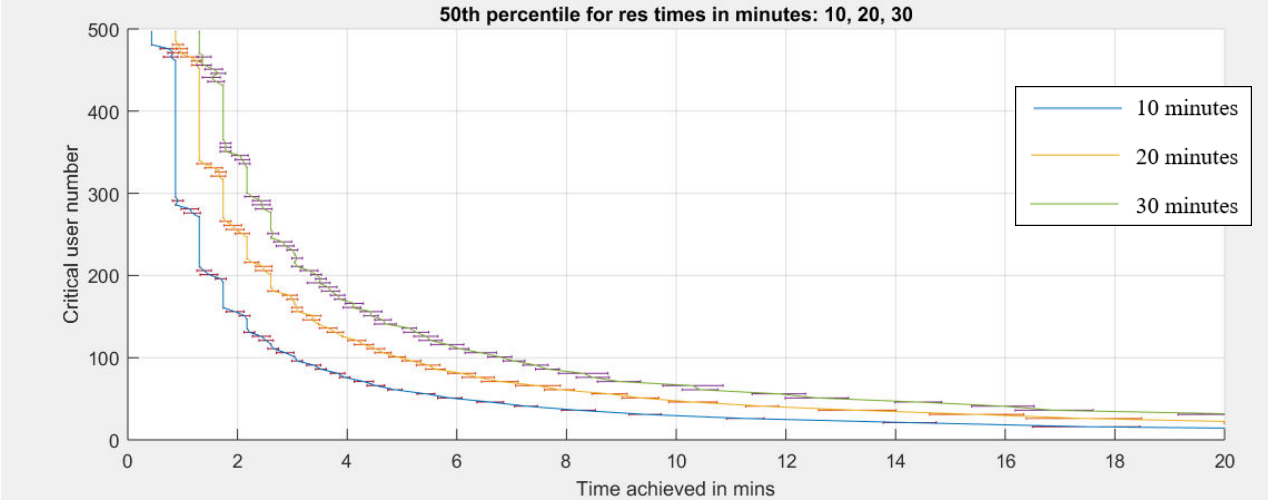

Figure 18: Critical mass of users vs. 50th percentile values of PoI sampling delay for variable mean residence times

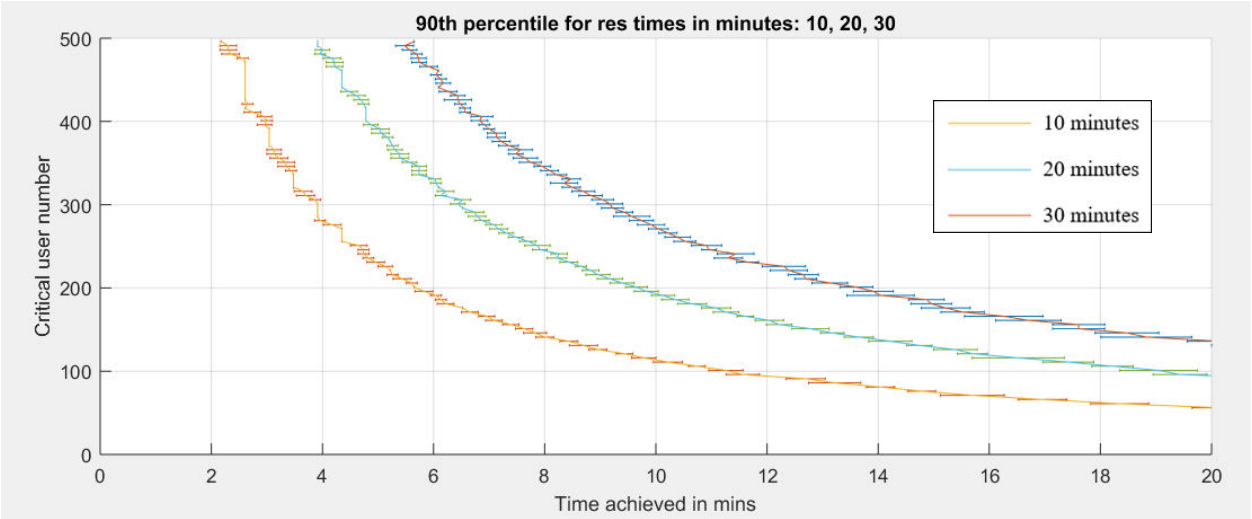

Figure 19: 90th percentile of application users' critical mass vs. PoI sampling delay for variable mean residence times

4.2 Impact of movement speed

Delay values are inversely related to pedestrian speed. The impact of this variable, however, seems to be less pronounced, compared to residence time, typically incurring delays less than a minute when it doubles, when dealing with more than 100 users. This happens for both 50th and 90th percentile cases. Speed also seems to matter less as it increases, something more evident in the 90th percentile distribution (figure 21).

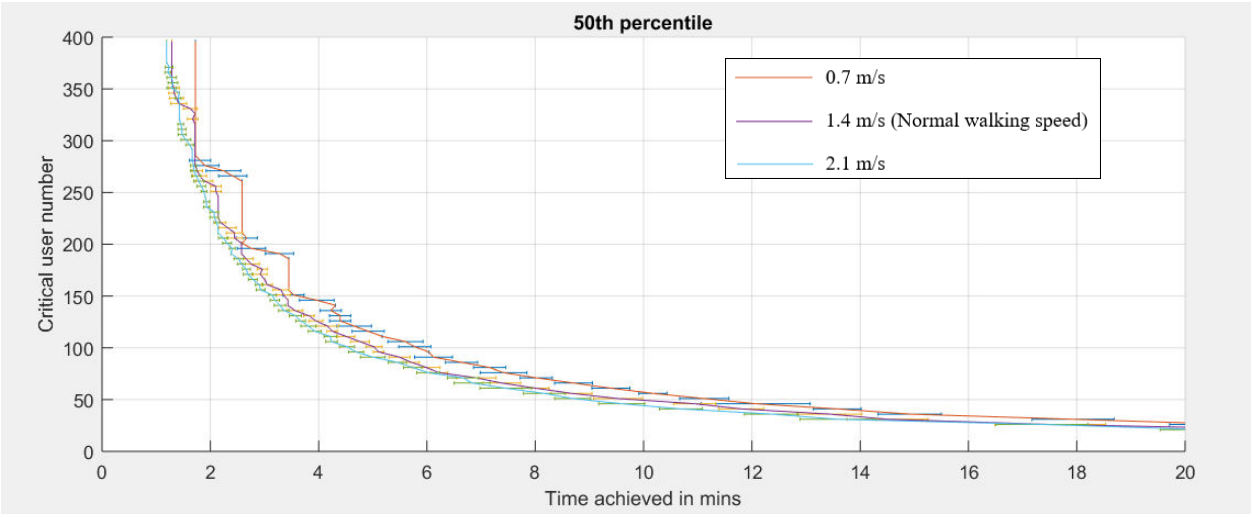

Figure 20: 50th percentile plot for variable movement speed

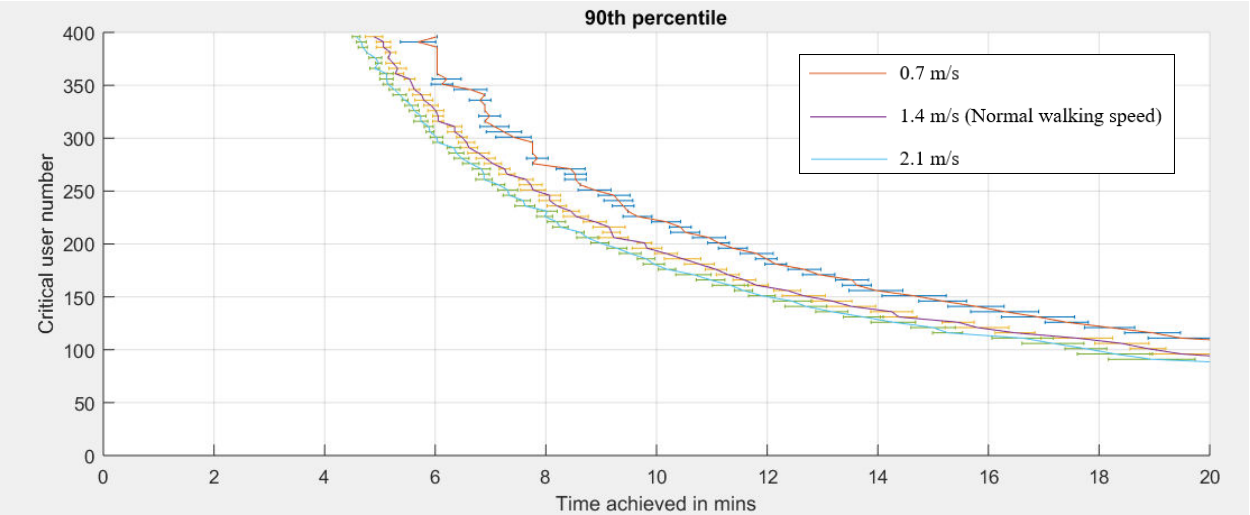

Figure 21: 90th percentile plot for variable movement speed

4.3 Hotspot destination probability

As mentioned, 18 cells were deemed to be the map "hotspots". Interestingly, for more users choosing a hotspot, 50th percentile delays drop, whereas 90th percentile delays increase. This could be attributed to some PoIs located near (or on) the hotspots being "favored" by such a scheme. For the rest of the PoIs however, increased probability leads to longer intervals between updates. The plots seem to indicate that the majority of PoIs experience better results with increased probability values, with delay values behaving non-linearly. It is also evident that "losers" suffer much more than "winners" gain when probability increases: for a typical gain of about 0.5 minutes in the 50th percentile, the 90th percentile delay may increase about 4 times as much.

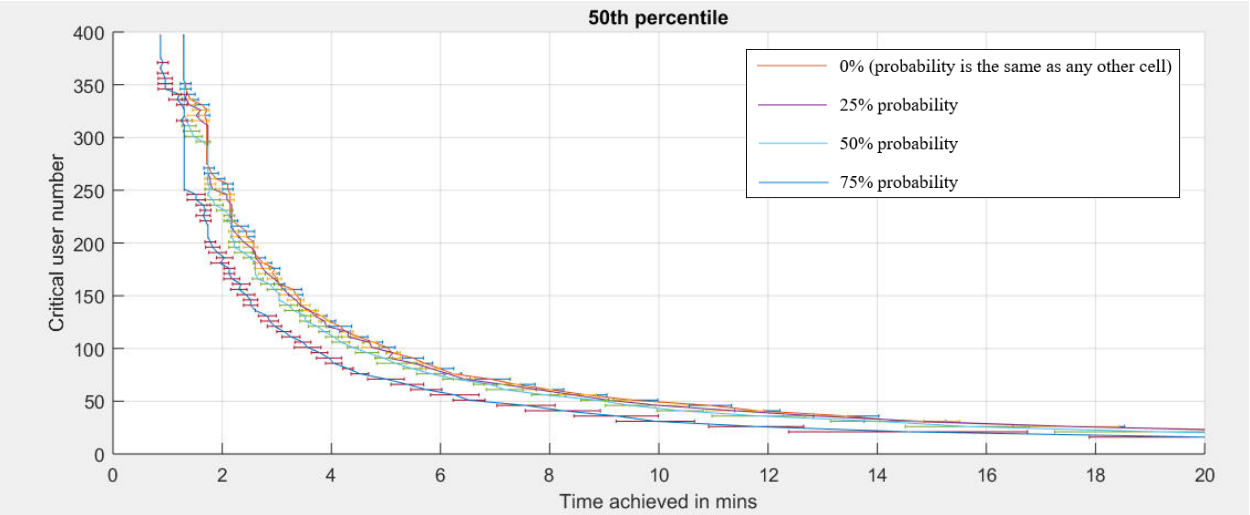

Figure 22: 50th percentile plot for variable hotspot probability

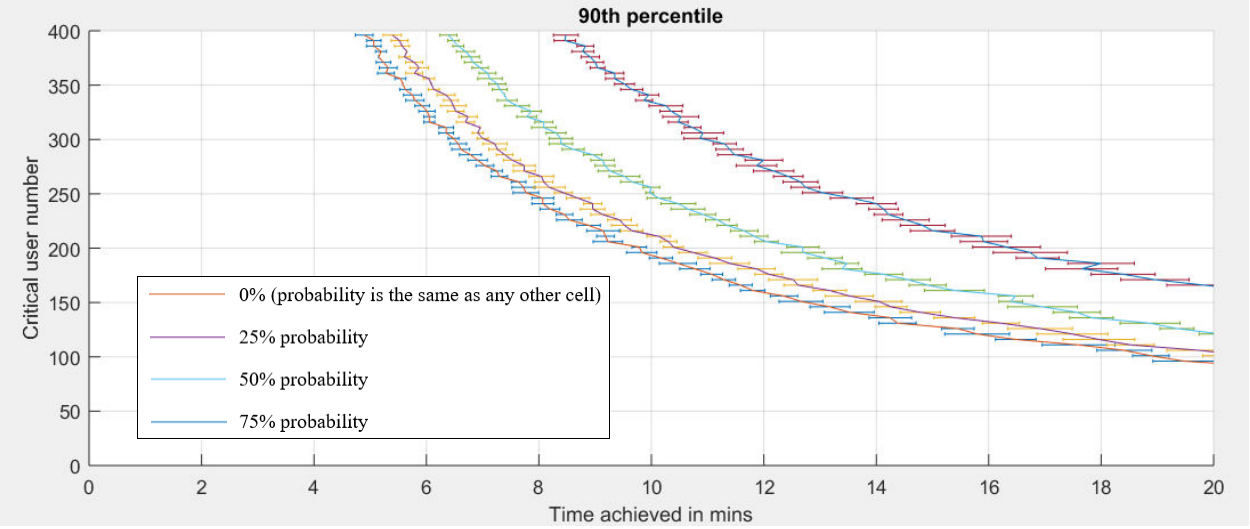

Figure 23: 90th percentile plot for variable hotspot probability

4.4 Impact of PoI reallocation

4.4.1 From center towards the periphery

As mentioned, the central zone is considered to be the central ¼ of the map. As probability for PoIs to relocate to this zone increases, delay numbers decrease, albeit

slightly. This can be attributed to the movement model used (Random Waypoint), which tends to favor the central area of a rectangular map [24]. The delay incurred seems to be fairly linear, although it tends to become more apparent for larger reallocation probability values. This becomes more pronounced when examining 90th percentile values, as shown in figure 25.

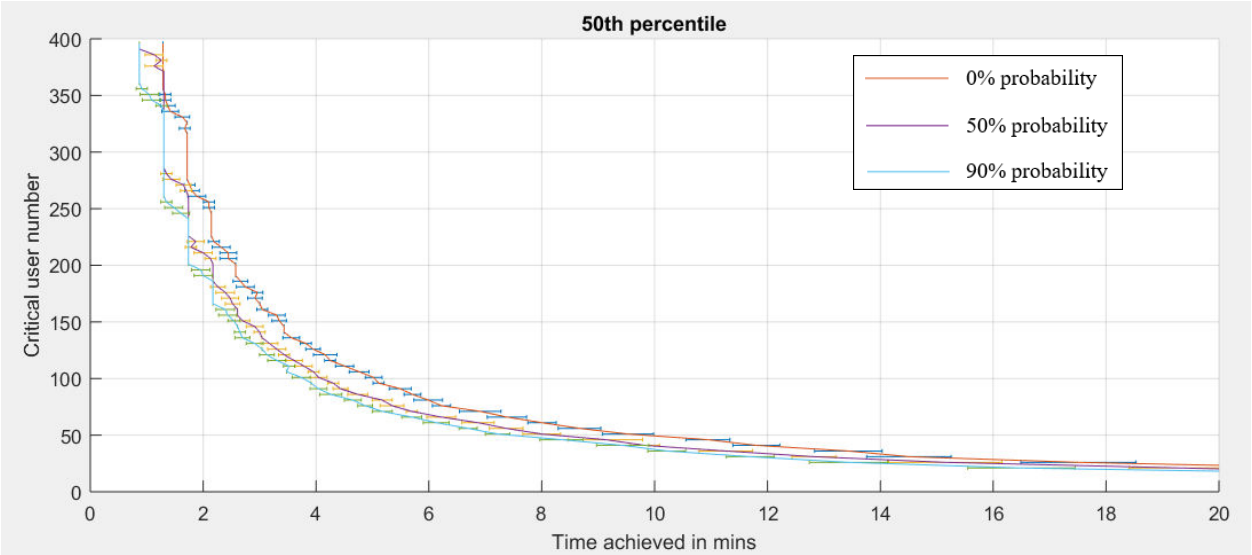

Figure 24: 50th percentile plot for variable central reallocation probability

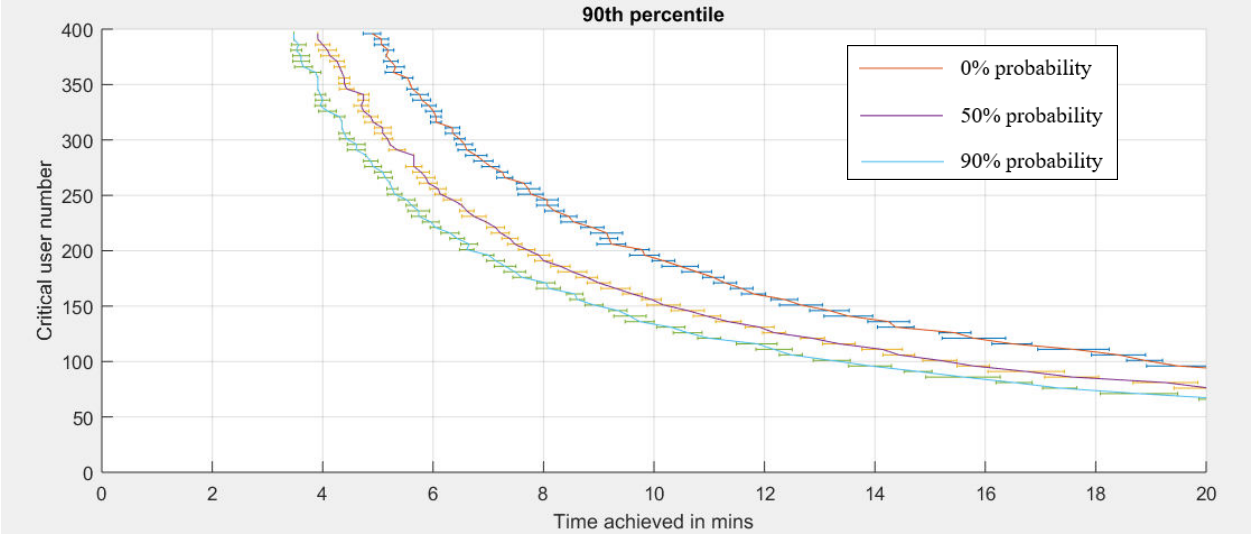

Figure 25: 90th percentile plot for variable central reallocation probability

4.4.2 From periphery to the center

As probability for PoIs to relocate to peripheral cells increases, delay values increase. This is the opposite from what happens in the previous example and, again, stems from the nature of the mobility model used. In this case however, values seem to behave more linearly, even for a large percentile distribution (figure 27).

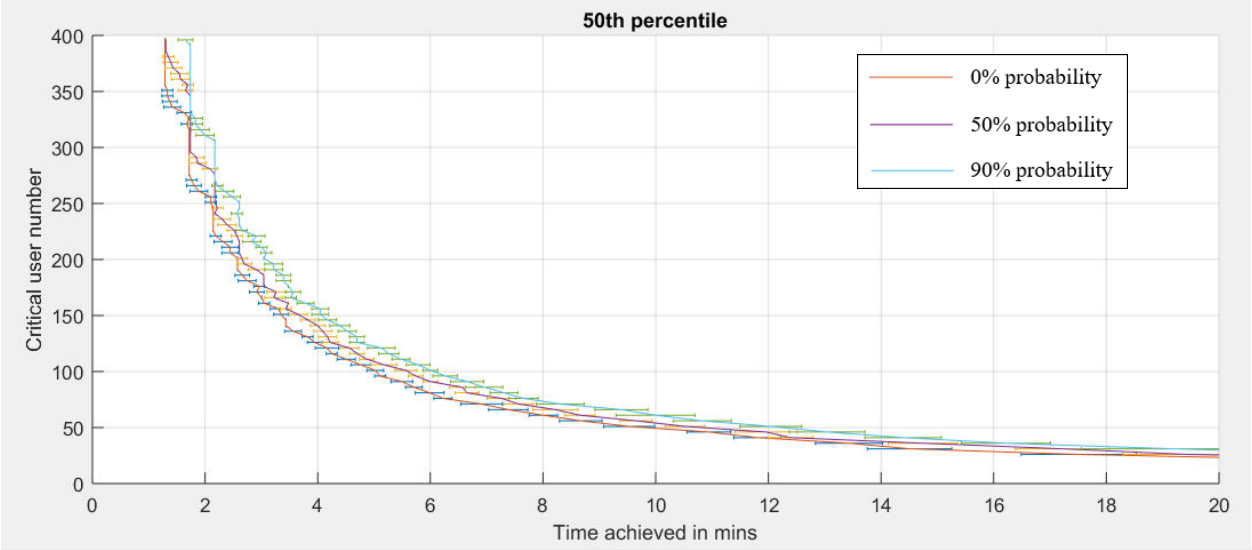

Figure 26: 50th percentile plot for variable peripheral reallocation probability

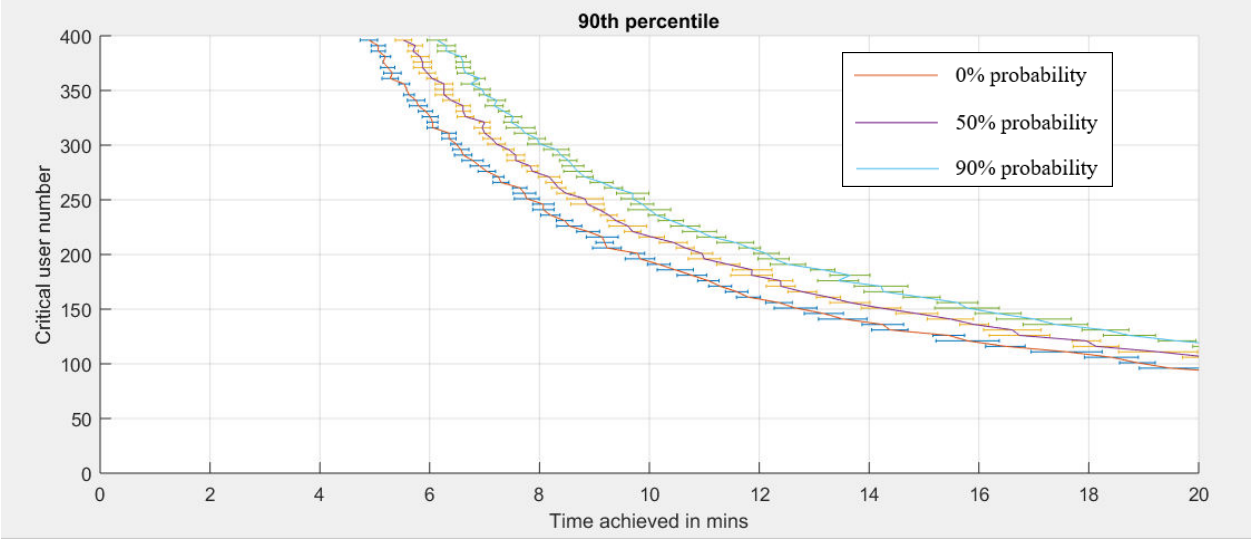

Figure 27: 90th percentile plot for variable peripheral reallocation probability

5 Conclusions

In this project, we have developed an approach towards evaluating potential crowdsourcing applications with regards to a key aspect – that of required user base. This approach assumes that performance can be depicted as frequency of user actions in certain places. In this context, a possible parking space finding application was evaluated. The user "action" here is sampling the status of a spot when (s)he sees it. Based on a map of Athens, we calculated how many users it would take to achieve a viable sampling delay, so that the application has fairly recent status information for all spots. Moreover, we examined the effect of various changes, be it in PoI topology or users' behavior.

Results for the map under investigation (Kolonaki-Athens) seem to indicate that under normal circumstances, no more than 150 users would be needed to support a 5 minute delay for 50% if time (a number possibly even lower for centrally located spots). This number would approximately double if we wanted to support the same delay in 90% of the times. Such user requirements could be deemed as easily achievable in a general fashion, but are obviously dependent on factors such as time of day, season, weather, etc. In what follows, we iterate on how this work could evolve towards a real-world tool with the flexibility to address different city environments and PoI distributions.

The simulator created for this project is flexible in its use potential. Here, the focus was on a specific area of Athens, for an application concerned with assessing the status of parking spots. In a similar fashion, maps of other areas can be used to evaluate user requirements for various crowdsourcing applications.

Given that the input of the simulator is a rectangular matrix of PoI numbers, any existing map/information would have to be converted to such a format. This can be done visually by overlaying a grid over the map and computing PoI numbers, as was the case in this project. However, the existence of digital maps would allow for the creation of a script to do the same job faster, and perhaps more accurately. Obviously, this requires the "open data" nature of the information, so the map files can be obtained, as well as examined to find out their structure.

Specifically in Greece, the GGRS87 [25] coordinate system is typically used in mapping, as is the case in Thessaloniki's largest mapping portal [26]. In this case

however, PoIs are supplied by a third-party company [27], which was inquired and turned out to hold its data as solely private.

Going one step further, by saving simulation output for different areas and forming a large enough repository, clustering/classification techniques could be utilized to identify trends based on factors such as population, street count, tourist attraction spots, etc. Besides saving processing time, this could also help when dealing with areas with no digital data available, or no "open data" policies. Instead, data readily available through more "traditional" means could be used to classify a new area in terms of user demands.

6 Bibliography

[1] http://www.crowdsourcing-blog.org/wp-content/uploads/2012/02/Towards-an-integrated-crowdsourcing-definition-Estell%C3%A9s-Gonz%C3%A1lez.pdf

[2] http://irl.cs.ucla.edu/~yefan/papers/IEEE-Com-Mag-11.pdf

[3]http://elearn.ihu.edu.gr/pluginfile.php/22204/mod_resource/content/1/Participatory_sensing.pdf

[4] Vazirani, Vijay V. (2001). *Approximation Algorithms*.

[5] Auer, S. R.; Bizer, C.; Kobilarov, G.; Lehmann, J.; Cyganiak, R.; Ives, Z. (2007). "DBpedia: A Nucleus for a Web of Open Data".

[6] https://opendata.ellak.gr/2015/07/23/crowdhackathon-transport-i-megaliteri-elliniki-drasi-kinonikis-ke-epichirimatikis-kenotomias-oloklirothike-me-epitichia/

[7] Hollands, R. G (2008). "Will the real smart city please stand up?"

[8] Komninos, Nicos (2013-08-22). "What makes cities intelligent?"

[9] https://www.mturk.com/mturk/findhits?match=false

[10] https://www.mturk.com/mturk/findquals?earned=false&requestable=false

[11] http://www.mturk-tracker.com/#/general

[12] http://www.wired.com/2011/07/mf_taskrabbit/

[13] http://index.okfn.org/dataset/

[14] http://www.popularmechanics.com/technology/infrastructure/a7437/smart-everything-even-lamp-posts-are-now-connected/

[15]http://elearn.ihu.edu.gr/pluginfile.php/21045/mod_resource/content/2/3.%20KDD%20Methodology%20IHU.pdf (page 35)

[16] http://www.theguardian.com/environment/2013/aug/06/fatberg-london-sewer-grease-blockage

[17] https://smartcity.wien.gv.at/site/en/projekte/verkehr-stadtentwicklung/

[18] https://www.wikipedia.org/

[19] https://www.openstreetmap.org/

[20] https://answers.yahoo.com/

[21] http://www.urbandictionary.com/

[22] http://www.mathworks.com/matlabcentral/fileexchange/22216-ploterr

[23] Browning, R. C., Baker, E. A., Herron, J. A. and Kram, R. (2006). "Effects of obesity and sex on the energetic cost and preferred speed of walking"

[24] http://data.bettstetter.com/publications/bettstetter-2002-wman-rwp.pdf

[25] http://mapref.org/CoordinateReferenceSystemsGR.html#Zweig743

[26] http://gis.thessaloniki.gr/gis2014/

[27] http://www.terra.gr/

[28] Broch, J.; Maltz, D. A.; Johnson, D. B.; Hu, Y. C.; Jetcheva, J. (1998). "A performance comparison of multi-hop wireless ad hoc network routing protocols".

[29] http://www.worldlibrary.org/articles/amazon_mechanical_turk

[30] J.-S. Lee and B. Hoh, “Dynamic Pricing Incentive for Participatory Sensing,” Pervasive Mobile Computing, vol. 6, no. 6, pp. 693–708, December 2010.

[31] L. Kazemi and C. Shahabi, “Geocrowd: Enabling query answering with spatial crowdsourcing,” in Proc. 20th ACM SIGSPATIAL GIS, 2012, pp. 189–198.

[32] Y. Ueyama, M. Tamai, Y. Arakawa, and K. Yasumoto, “Gamification-based incentive mechanism for participatory sensing,” in Workshops IEEE PERCOM ’14, March 2014, pp. 98–10

[33] W. Mason and D. J.Watts, “Financial Incentives and the “Performance of Crowds”,” in Proc. ACM SIGKDD on Human Computation (HCOMP), 2009, pp. 77–85.

[34] X. S. Yang, D. W. Cheung, L. Mo, R. Cheng, and B. Kao, “On incentive-based tagging,” in Proc. IEEE ICDE ’13, 2013, pp. 685–696.

[35] A. Singla and A. Krause, “Truthful Incentives in Crowdsourcing Tasks Using Regret Minimization Mechanisms,” in Proc. 22nd International Conference on World Wide Web (WWW) , 2013, pp. 1167–1178.

Merkouris Karaliopoulos

Digitally signed by Merkouris Karaliopoulos
Date: 2016.05.10 10:48:34 +03'00'